

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: Линейные структуры данных: стек, очередь, дек
Вариант 11-д

Студент гр. 8304

Щука А. А.

Преподаватель

Фирсов М. А.

Санкт-Петербург

2019

Цель работы.

Познакомиться с часто используемыми на практике линейными структурами данных, обеспечивающими доступ к элементам последовательности только через её начало и конец, и способами реализации этих структур, освоить на практике использование стека, очереди и дека для решения задач.

Постановка задачи.

Рассматривается выражение следующего вида:

$$\begin{aligned} < \text{выражение} > ::= < \text{терм} > \mid < \text{терм} > + < \text{выражение} > \mid \\ & \qquad \qquad \qquad < \text{терм} > - < \text{выражение} > \\ < \text{терм} > ::= < \text{множитель} > \mid < \text{множитель} > * < \text{терм} > \\ < \text{множитель} > ::= < \text{число} > \mid < \text{переменная} > \mid (< \text{выражение} >) \mid \\ & \qquad \qquad \qquad < \text{множитель} > ^ < \text{число} > \\ < \text{число} > ::= < \text{цифра} > \\ < \text{переменная} > ::= < \text{буква} > \end{aligned}$$

Такая форма записи выражения называется *инфиксной*.

Постфиксной (префиксной) формой записи выражения aDb называется запись, в которой знак операции размещен за (перед) операндами: abD (Dab).

Примеры

<i>Инфиксная</i>	<i>Постфиксная</i>	<i>Префиксная</i>
$a-b$	$ab-$	$-ab$
$a*b+c$	$ab*c+$	$+*abc$
$a*(b+c)$	$abc+*$	$*a+bc$
$a+b^c^d*e$	abc^d^e*+	$+a*^b^cde.$

Отметим, что постфиксная и префиксная формы записи выражений не содержат скобок.

Вариант 11-д: вывести в обычной (инфиксной) форме выражение, записанное в постфиксной форме в заданном текстовом файле postfix (рекурсивные процедуры не использовать и лишние скобки не выводить);

Описание алгоритма.

Для начала программа должна считать данные и передать строку в функцию преобразования выражения из постфиксной формы в инфиксную. Далее для перевода выражения из постфиксной в инфиксную запись необходимо следовать алгоритму:

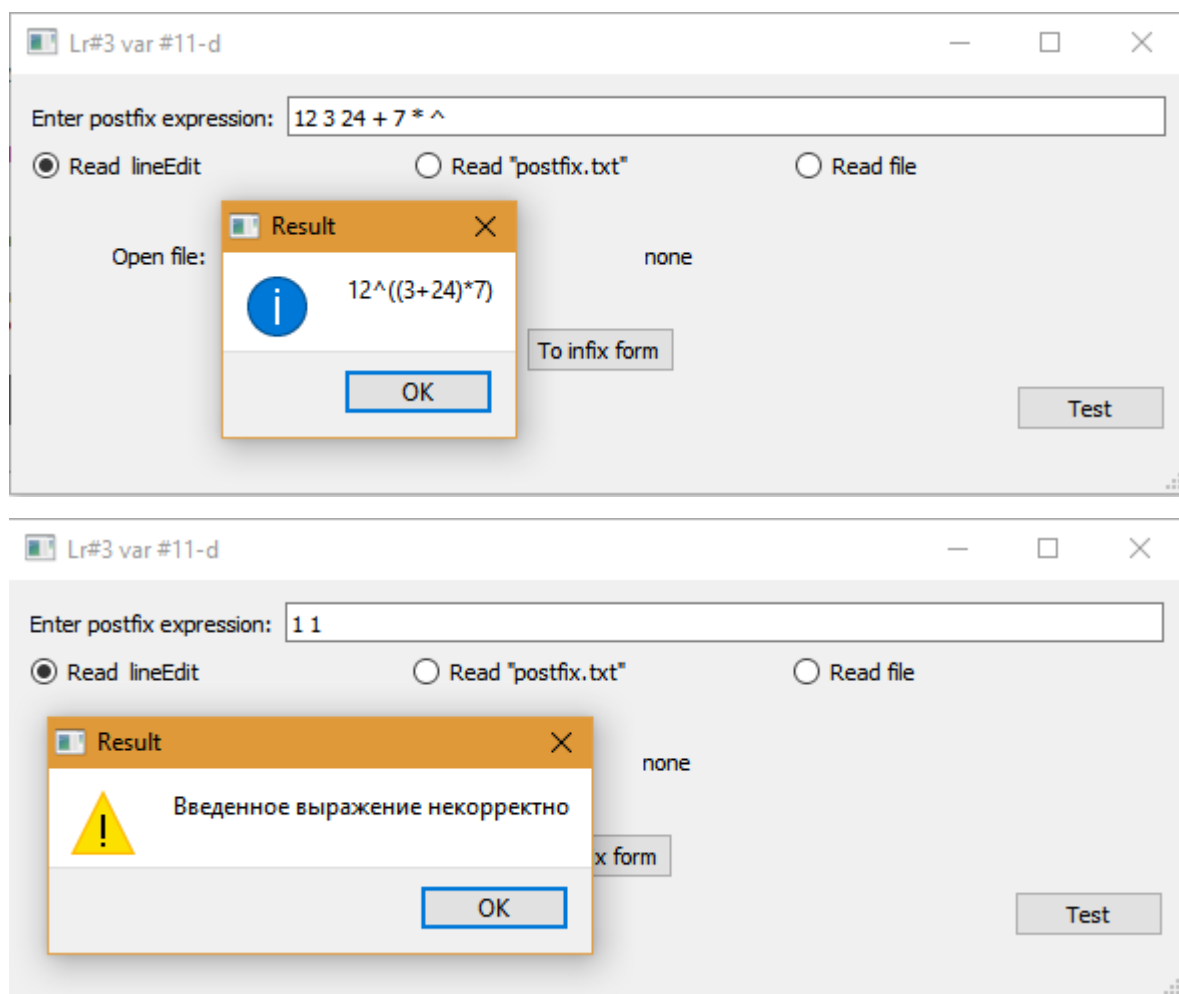
- 1) Если читаем число, то заносим его в стек
- 2) Если читаем знак операции, то:
 - а) Берем текущий знак операции и следующий
 - б) Если в первом элементе приоритет операции меньше (и не равен 0), чем у рассматриваем операции, то берем первый элемент в скобки
 - в) Аналогично для 2-го элемента
 - г) Записываем в стек строку вида: 2-й элемент + знак операции + 1-й элемент
- 3) Если строка полностью пройдена, то результатом является значение вершины стека

Спецификация программы.

Программа предназначена для перевода записи выражения из постфиксной в инфиксную.

Программа написана на языке C++ с использованием фреймворка Qt. Входными данными являются цифры и знаки арифметических операций, считываемые из файла postfix.txt, дополнительно реализована возможность ввода с помощью GUI. Выходными данными являются промежуточные значения и конечный результат. Данные выводятся в QMessageBox.

Рисунок 1- Результат работы программы



Тестирование.

Таблица 1 – Результаты тестирования программы

Input1	Output
1 2 +	2+1
1 2 3 + *	(3+2)*1
2 3 4 5 + * +	(5+4)*3+2
1 2 3 4 5 + + * +	(5+4+3)*2+1
1 2 * 3 +	2*1+3
1 2 3 4 + 5 + * *	(4+3+5)*2*1
1 2 3 4 ^ - *	1*(2-3^4)
12 * 1	Введенное выражение некорректно

17	17
$r t + y * 12 ^$	$((r+t)*y)^{12}$
+	Введенное выражение некорректно
1 1 1 +	Введенное выражение некорректно

Анализ алгоритма.

Алгоритм работает за линейное время от размера строки. Для экономии памяти строка передается по константной ссылке.

Описание функций и СД.

Класс MyStack реализует структуру стека, а также методы для работы с ним.

Стандартные методы для работы со стеком:

```
void pop();
void push(const Node& elem);
bool isEmpty() const;
Node top() const;
size_t size() const;
```

Класс Mainwindow реализует алгоритм перевода выражения из постфиксной формы в инфиксную с помощью стека.

Статический метод для перевода выражения из постфиксной формы в инфиксную:

```
static std::string toInfix(const std::string& expression);
```

Принимает на вход константную ссылку на строку-выражение, возвращает выражение в инфиксной форме, если исходное выражение корректно и пустую строку в случае ошибки. Строка анализируется посимвольно. Если текущий символ "число или буква", тогда элемент помещается в стек с приоритетом 0, если текущий символ "знак операции", из стека достаются два элемента, записываются в временную строку с учетом приоритетов операций и временная

строка помещается в стек с приоритетом знака. В ходе преобразования, если стек пустой, выводится ошибка и возвращается пустая строка. После преобразования в стеке должен находиться один элемент - инфиксное выражение.

Выводы.

В ходе работы были приобретены навыки работы со стеком, изучены методы работы с ним (объявлять, заносить в него переменных и забирать их). Был изучен и реализован алгоритм перевода записи из постфиксной в инфиксную.

Приложение А. Исходный код программы.

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#define QT_NO_DEBUG_OUTPUT

#include <QMainWindow>
#include <QTextStream>
#include <QFile>
#include <QFileDialog>
```

```

#include <QMessageBox>
#include <QDebug>
#include <QString>
#include <QDir>
#include <QStringList>
#include <string>
#include "mystack.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

public slots:
    static std::string toInfix(MyStack* stack, const std::string& expression);
    static bool isSign(const char ch);
    static bool isAlpha(const char ch);
    static bool isDigit(const char ch);

private slots:
    void on_readFile_radioButton_clicked();

    void on_readLineEdit_radioButton_clicked();

    void on_readPostfix_radioButton_clicked();

    void on_toInfix_pushButton_clicked();

private:
    Ui::MainWindow *ui;
    QTextStream* in;
    QFile* file;
    QDir* dir;
};

#endif // MAINWINDOW_H

```

mystack.h

```

#ifndef MYSTACK_H
#define MYSTACK_H

#define QT_NO_DEBUG_OUTPUT

#include <QObject>
#include <string>
#include <QDebug>
#include <memory>

//перечисления для приоритета операций
enum class optype {power = 3, multiply = 2, minus = 1, plus = 1, null = 0};

//Элемент стека, состоящий из выражения и приоритета этого выражения
typedef std::pair<std::string, optype> Data;

class MyStack : public QObject
{
    /*
     * Класс-реализация стека, для перевода выражения из
     * постфиксной формы в инфиксную.
     */

    Q_OBJECT

public:
    struct Node{
        typedef std::shared_ptr<Node> NodeP;

```

```

        Data elem;
        NodeP nextNode;
    };

    explicit MyStack(QObject* parent = nullptr);
    ~MyStack() = default;

    //Запрет конструктора копирования и оператора присваивания
    MyStack& operator=(const MyStack& list) = delete;
    MyStack(const MyStack& list) = delete;

public slots:
    //Стандартные методы для работы со стеком
    void pop();
    void push(const Data elem);
    bool isEmpty() const;
    Data top() const;
    size_t size() const;

private:
    Node::NodeP topElem;
    size_t sizeStack;
};

#endif // MYSTACK_H

```

main.cpp

```

#include "mainwindow.h"
#include <QApplication>

//var #11-d

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    MainWindow window;
    window.setWindowTitle("Lr#3 var #11-d");
    window.show();

    return app.exec();
}

```

mainwindow.cpp

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    file = new QFile;
    in = new QTextStream;
    dir = new QDir;

    ui->readPostfix_radioButton->setChecked(true);
    emit on_readPostfix_radioButton_clicked();
}

MainWindow::~MainWindow()
{
    delete ui;
    delete in;
    delete file;
    delete dir;
}

void MainWindow::on_readFile_radioButton_clicked()

```



```

{
    //выбор считывания из файла или из окна
    file->close();

    ui->textLabel_1->setEnabled(false);
    ui->input_lineEdit->setEnabled(false);
    QString fileName = QFileDialog::getOpenFileName(this,
                                                    "Open file");

    file->setFileName(fileName);
    file->open(QFile::ReadOnly | QFile::Text);
    in->setDevice(file);

    ui->openFile_textLabel->setText(fileName);
}

void MainWindow::on_readLineEdit_radioButton_clicked()
{
    file->close();

    ui->input_lineEdit->setEnabled(true);
    ui->textLabel_1->setEnabled(true);
    ui->openFile_textLabel->setText("none");
}

void MainWindow::on_readPostfix_radioButton_clicked()
{
    file->close();

    ui->input_lineEdit->clear();
    ui->input_lineEdit->setEnabled(false);
    ui->textLabel_1->setEnabled(false);

    QString fileName = QDir::currentPath() + "/Tests/postfix.txt";
    file->setFileName(fileName);
    file->open(QFile::ReadOnly | QFile::Text);
    in->setDevice(file);

    ui->openFile_textLabel->setText(fileName);
}

void MainWindow::on_toInfix_pushButton_clicked()
{
    qDebug();
    qDebug() << "Преобразование выражения из постфиксной в инфиксную форму:";

    //строка для считанного результата
    std::string tmpStr = "";

    //считывание строки из файла или из окна
    if (ui->readFile_radioButton->isChecked() ||
        ui->readPostfix_radioButton->isChecked()) {
        tmpStr = in->readLine().toStdString();
    }
    else {
        tmpStr = ui->input_lineEdit->text().toStdString();
    }

    qDebug() << "__Считанное выражение__:" << tmpStr.c_str();

    //Преобразование выражения в инфиксную форму
    MyStack *stack = new MyStack;
    std::string result = MainWindow::toInfix(stack, tmpStr);
    if (result != "") {
        qDebug() << "Выражение в инфиксной форме: " << result.c_str();
        QMessageBox::information(this, "Result", result.c_str());
    }
    else {
        qDebug() << "Введенное выражение некорректно";
        QMessageBox::warning(this, "Result", "Введенное выражение некорректно");
    }
    delete stack;

    ui->input_lineEdit->clear();
    ui->readFile_radioButton->setChecked(false);
    ui->readLineEdit_radioButton->setChecked(false);
    ui->readPostfix_radioButton->setChecked(true);
}

```

```

    ui->textLabel_1->setEnabled(false);
    ui->input_lineEdit->setEnabled(false);
    emit on_readPostfix_radioButton_clicked();
}

std::string MainWindow::toInfix(MyStack* stack, const std::string &expression)
{
    /*
     * Статический метод класса, для преобразования выражения,
     * заданного в постфиксной форме, в инфиксную.
     *
     * Принимает на вход константную ссылку на строку-выражение, возвращает
     * выражение в инфиксной форме, если исходное выражение корректно и пустую
     * строку в случае ошибки.
     *
     * Строка анализируется посимвольно. Если текущий символ "число или буква", тогда
     * элемент помещается в стек с приоритетом 0, если текущий символ "знак операции",
     * из стека достаются два элемента, записываются в временную строку с учетом приоритетов
     * операций и временная строка помещается в стек с приоритетом знака.
     *
     * В ходе преобразования, если стек пустой, выводится ошибка и возвращается пустая
     * строка. После преобразования в стеке должен находиться один элемент - инфиксное выражение.
     */

    std::string tmpStr = "";

    for (auto i = expression.cbegin(); i < expression.end(); ++i) {
        char elem = *i;
        if (elem == ' ') {
            continue;
        }
        else if (isDigit(elem) || isAlpha(elem)) {
            bool isDigitElem = isDigit(elem);
            while (elem != ' ' && i != expression.end()) {
                if ((isDigitElem && isAlpha(elem)) ||
                    (!isDigitElem && !isAlpha(elem))) {
                    qDebug() << "Error: wrong data!";
                    return "";
                }
                tmpStr += elem;
                ++i;
                elem = *i;
            }
            stack->push(Data(tmpStr, otype::null));

            qDebug() << "В стек помещается элемент:" << elem;
        }
        else if (isSign(elem)) {
            Data firstArg;
            Data secondArg;

            qDebug() << "Знак операции:" << elem;

            if (!stack->isEmpty()) {
                secondArg = stack->top();
                stack->pop();
            }
            else {
                qDebug() << "Error: stack is empty!";
                return "";
            }

            if (!stack->isEmpty()) {
                firstArg = stack->top();
                stack->pop();
            }
            else {
                qDebug() << "Error: stack is empty!";
                return "";
            }

            qDebug() << "Выражения в стеке: " << firstArg.first.c_str() <<
                "и" << secondArg.first.c_str();

            otype tmpOtype = otype::null;
            if (elem == '+') {
                tmpOtype = otype::plus;
            }
            else if (elem == '-') {
                tmpOtype = otype::minus;
            }
        }
    }
}

```

```

    } else if (elem == '*') {
        tmpOptype = optype::multiply;
    } else if (elem == '^') {
        tmpOptype = optype::power;
    }

    if (firstArg.second != optype::null && firstArg.second < tmpOptype) {
        tmpStr += '(';
        tmpStr += firstArg.first;
        tmpStr += ')';
    } else {
        tmpStr += firstArg.first;
    }

    tmpStr += elem;

    if (secondArg.second != optype::null && secondArg.second < tmpOptype) {
        tmpStr += '(';
        tmpStr += secondArg.first;
        tmpStr += ')';
    } else {
        tmpStr += secondArg.first;
    }

    qDebug() << "Итоговое выражение помещается в стек:" << tmpStr.c_str();

    stack->push(Data(tmpStr, tmpOptype));
}
else {
    qDebug() << "Некорректный символ в строке!";
    return "";
}
tmpStr = "";
}

if (stack->size() == 1) {
    tmpStr = stack->top().first;
    stack->pop();
    return tmpStr;
}
else {
    qDebug() << "Error: string is incorrect!";
    return "";
}
}

bool MainWindow::isDigit(const char ch)
{
    /*
     * Проверка на цифру
     */

    return (ch >= '0' && ch <= '9');
}

bool MainWindow::isAlpha(const char ch)
{
    /*
     * Проверка на букву
     */

    return ((ch >= 'a' && ch <= 'z') ||
            (ch >= 'A' && ch <= 'Z'));
}

bool MainWindow::isSign(const char ch)
{
    /*
     * Проверка на знак
     */

    return (ch == '+' || ch == '-' ||
            ch == '*' || ch == '^');
}

```

mystack.cpp

```
#include "mystack.h"

MyStack::MyStack(QObject* parent) : QObject(parent)
{
    topElem = nullptr;
    sizeStack = 0;
}

void MyStack::pop()
{
    /*
     * Удаление верхнего элемента из стека
     */
    if (topElem != nullptr) {
        Node::NodeP oldTopElem = topElem;
        topElem = oldTopElem->nextNode;
        sizeStack -= 1;
        oldTopElem->nextNode = nullptr;
    }
    else {
        qDebug() << "Error: stack is empty";
    }
}

void MyStack::push(const Data elem)
{
    /*
     * Добавление элемента в стек
     */

    Node::NodeP tmp = topElem;
    topElem = Node::NodeP(new Node);
    topElem->elem = elem;
    sizeStack += 1;
    topElem->nextNode = tmp;
}

bool MyStack::isEmpty() const
{
    /*
     * Проверка стека на пустоту
     */

    return topElem == nullptr;
}

Data MyStack::top() const
{
    /*
     * Доступ к верхнему элементу стека
     */
    if (topElem != nullptr) {
        return topElem->elem;
    }
    else {
        qDebug() << "Error: stack is empty!";
        return Data("", optype::null);
    }
}

size_t MyStack::size() const
{
    /*
     * Получение размера стека
     */

    return sizeStack;
}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>MainWindow</class>
  <widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>400</width>
        <height>209</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>MainWindow</string>
    </property>
    <widget class="QWidget" name="centralWidget">
      <layout class="QVBoxLayout" name="verticalLayout_2">
        <item>
          <layout class="QHBoxLayout" name="horizontalLayout">
            <item>
              <widget class="QLabel" name="textLabel_1">
                <property name="text">
                  <string>Enter postfix expression:</string>
                </property>
              </widget>
            </item>
            <item>
              <widget class="QLineEdit" name="input_lineEdit"/>
            </item>
          </layout>
        </item>
        <item>
          <layout class="QHBoxLayout" name="horizontalLayout_2">
            <item>
              <widget class="QRadioButton" name="readLineEdit_radioButton">
                <property name="text">
                  <string>Read lineEdit</string>
                </property>
              </widget>
            </item>
            <item>
              <widget class="QRadioButton" name="readPostfix_radioButton">
                <property name="text">
                  <string>Read &quot;postfix.txt&quot;</string>
                </property>
              </widget>
            </item>
            <item>
              <widget class="QRadioButton" name="readFile_radioButton">
                <property name="text">
                  <string>Read file</string>
                </property>
              </widget>
            </item>
          </layout>
        </item>
        <item>
          <layout class="QHBoxLayout" name="horizontalLayout_5">
            <item>
              <spacer name="horizontalSpacer_2">
                <property name="orientation">
                  <enum>Qt::Horizontal</enum>
                </property>
                <property name="sizeType">
                  <enum>QSizePolicy::Minimum</enum>
                </property>
                <property name="sizeHint" stdset="0">
                  <size>
                    <width>40</width>
                    <height>20</height>
                  </size>
                </property>
              </spacer>
            </item>
            <item>
              <widget class="QLabel" name="textLabel_0">

```

```

        <property name="text">
            <string>Open file:</string>
        </property>
    </widget>
</item>
<item>
    <widget class="QLabel" name="openFile_textLabel">
        <property name="text">
            <string>Label</string>
        </property>
    </widget>
</item>
</layout>
</item>
<item>
    <layout class="QHBoxLayout" name="horizontalLayout_4">
        <item>
            <spacer name="horizontalSpacer_3">
                <property name="orientation">
                    <enum>Qt::Horizontal</enum>
                </property>
                <property name="sizeHint" stdset="0">
                    <size>
                        <width>40</width>
                        <height>20</height>
                    </size>
                </property>
            </spacer>
        </item>
        <item>
            <widget class="QPushButton" name="toInfix_pushButton">
                <property name="text">
                    <string>To infix form</string>
                </property>
            </widget>
        </item>
        <item>
            <spacer name="horizontalSpacer_4">
                <property name="orientation">
                    <enum>Qt::Horizontal</enum>
                </property>
                <property name="sizeHint" stdset="0">
                    <size>
                        <width>40</width>
                        <height>20</height>
                    </size>
                </property>
            </spacer>
        </item>
    </layout>
</item>
<item>
    <layout class="QHBoxLayout" name="horizontalLayout_3">
        <item>
            <spacer name="horizontalSpacer">
                <property name="orientation">
                    <enum>Qt::Horizontal</enum>
                </property>
                <property name="sizeHint" stdset="0">
                    <size>
                        <width>40</width>
                        <height>20</height>
                    </size>
                </property>
            </spacer>
        </item>
        <item>
            <widget class="QPushButton" name="test_pushButton">
                <property name="text">
                    <string>Test</string>
                </property>
            </widget>
        </item>
    </layout>
</item>
</layout>
</widget>
<widget class="QMenuBar" name="menuBar">
    <property name="geometry">

```

```
<rect>
  <x>0</x>
  <y>0</y>
  <width>400</width>
  <height>21</height>
</rect>
</property>
</widget>
<widget class="QStatusBar" name="statusBar"/>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections/>
</ui>
```